Technical Disclosure Commons

Defensive Publications Series

October 2020

STATEFUL CONTENT SELECTION

STATEFUL CONTENT SELECTION

When a client device renders a website provided by a content provider and executes the code of the website, the code can cause small amounts of data to be stored on the client device (e.g., a Hypertext Transfer Protocol (HTTP) cookie). The data can include stateful information about, for example, preferences or previous interactions with the content provider, or information about the client device (e.g., device identifiers, user accounts, or any other such information). In some implementations, the data stored in the HTTP cookie can enable the content provider to select content that is customized for the client device. In some implementations, the data can include an authentication token that informs the content provider that the client device was previously authenticated by the content provider. The presence of the authentication token on the client device can mean the client device does not need to re-authenticate with the content provider (e.g., provide a user name and password again) to receive content from the content provider the next time the client device requests content from the content provider. Not having to re-authenticate with the content provider can save time and bandwidth as the client device does not have to go through the authentication processes each time the client device interacts with the content provider for additional data.

In some implementations, the web browser can partition the HTTP cookie at a predetermined amount of time (e.g., 24 hours) after the HTTP cookie is stored on the client device. Prior to being partitioned, the content provider that caused the HTTP cookie to be stored on the client device can access the HTTP cookie regardless of the website the client device is visiting. For example, the HTTP cookie can be accessed in both a third-party context and a first-party context by the content provider that caused the HTTP cookie to be stored on the client device. In some implementations, once partitioned, the content provider that caused the HTTP

cookie to be stored on the client device cannot access the data of the HTTP cookie when the client device is visiting a third-party website. Rather than accessing the original HTTP cookie, once partitioned, and in a third-party context, the content provider can store new HTTP cookies in unique, isolated storage. When the client device re-visits a website provided by the content provider (e.g., the client device visits the website in a first-party context again), the HTTP cookie can temporarily become unpartitioned for the predetermined amount of time. In some implementations, HTTP cookies generated in the unique, isolated storage can be deleted when the HTTP cookie is unpartitioned. “Partitioning” may refer to the process of “sandboxing”, “walling off”, “locking”, or “making private” an HTTP cookie.

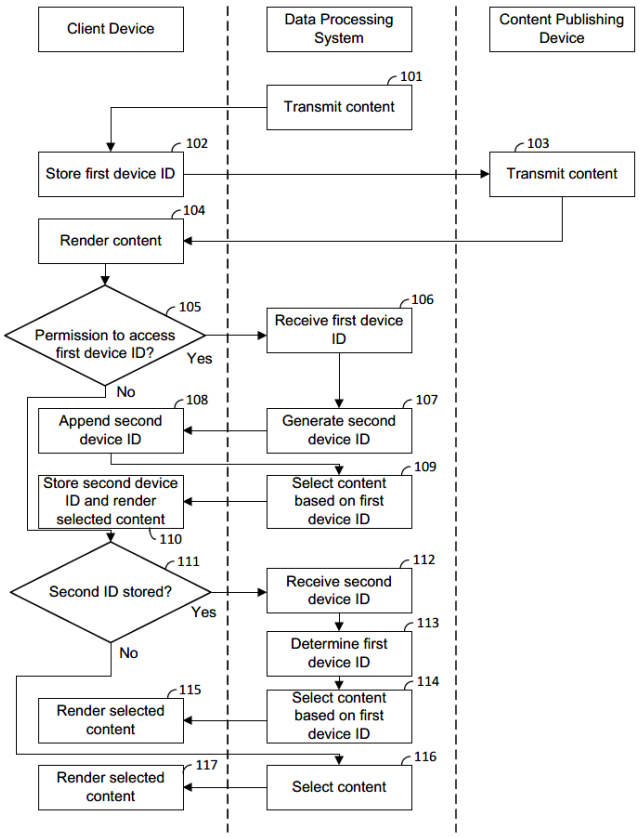
In some implementations, after the predetermined amount of time, the content provider can no longer access data on the client device that it caused to be stored there because the web browser includes a policy for limiting access. Additionally, some countries have implemented regulations that can prevent content providers from placing or accessing third-party cookies, and these regulations may be enforced by the browser.

However, in some implementations, a first content provider that the web browser previously visited can also provide content to the web browser via a second content provider. For example, the first content provider may host videos that can be embedded within the web pages of the second content provider. The user of the web browser may wish for the content provided via the second content provider (by the first content provider) to be selected according to a previous session’s data with the first content provider. For example, accessing the content may require authentication with the first content provider or selection based on user preferences. However, the session data (e.g., client device identifiers associated with the preferences) may not be available to the second content provider because the session data was stored on the client

device as an HTTP cookie by the first content provider. As described above, the web browser can limit or prevent the second content provider's access to HTTP cookies of the first content provider. Additionally, once the HTTP cookie is partitioned, the first content provider also may not be able to access the HTTP cookie to provide content via the second content provider's website. This can prevent single sign-on authentication or the selection of customized content. In this example, because the content selection is processed via the domain of the first content provider that the web browser is not directly visiting, the web browser can prevent the first content provider from accessing the unpartitioned HTTP cookies of the first content provider. This can prevent the first content provider from selecting customized content for inclusion in the primary content of the second content provider.

In some implementations, through limiting cookies to first-party cookies only, the systems and methods described herein can enable the caching of the data within an HTTP cookie from a first website into the HTTP cookie of a second website. Thus, the systems and methods described herein enable the exchange of information between third-party entities that would be blocked by typical web browsers and other computing devices that prevent cross-domain data exchange. This allows for rich, multi-source content (e.g., web pages (also referred to as primary content items) with embedded third-party content, mixed media, or other such content, which can generally be referred to as content items, secondary content items, or digital components), without sacrificing security of the client device and browser environment.

A swim lane diagram illustrating an implementation of these systems and methods is provided and discussed below:



At step 101, a data processing system can transmit content to the client device. The data processing system may do so upon receiving a request for content from the client device. For example, a web browser executed by the client device can request content from the data processing system, which can also function as a content publishing device. The response can be an HTTP request and the content can be primary content (e.g., a web page, data file, media file, or other such content). The primary content can include text, executable instructions (such as JavaScript), and other content.

At step 102, the client device can receive the primary content and a first device ID. The primary content can include executable scripts that can cause the client computing device to store the first device ID at the client computing device when rendering the primary content. The first device ID can be stored as or in an HTTP cookie, or more generally a data file, on the client

device. The client device may render the content on the web page. In some implementations, the web browser can set a time at which the data file expires or the browser partitions the data file.

At step 103 a content publishing device can transmit primary content to the client device. The primary content can include one or more content slots for secondary content, sometimes referred to as embedded content (e.g., images, media, executable scripts, etc., provided from another source or server). At step 104, the web browser can render the primary content received from the content publishing device. At step 105, the client device can determine if the primary content provided by the content publishing device has permission to access the first device ID. As described above, the first device ID may be stored in an HTTP cookie that is associated with the data processing system. The data processing system can provide executable scripts to the content publishing device, which may be included in the primary content transmitted to the client device at step 103. In some implementations, the browser can partition HTTP cookies after a predetermined period of time. Once partitioned, the executable scripts provided by the data processing system in the primary content may no longer have access to the data processing system's HTTP cookie. For example, once partitioned the HTTP cookie (and device IDs contained therein) may not be available in a third-party context.

If at step 105 the browser is able to access the first device ID, at step 106, the browser can transmit the first device ID to the data processing system. The data processing system can receive the first device ID and, at step 107, the data processing system can generate the second device ID based on the first device ID. For example, the data processing system may generate the second device ID by encrypting or hashing the first device ID. In some implementations, the second device ID can be generated at the client device. The data processing system may transmit the second device ID to the client device.

Once the client device receives or generates the second device ID, at step 108, the client device can append the second device ID to one or more content requests. The content requests can be for additional content to fill the content slots of the primary content item received at step 103.

At step 109, the data processing system can select content based on the first device ID. For example, the data processing system can perform or be associated with a content or ad serving system. The data processing system can use the second device ID to determine or lookup the first device ID. The first device ID can be associated with content selection preferences, authentication tokens, or other parameters that the data processing system can use to make content selections.

At step 110, the client device can store the second device ID and render the selected content. The browser can receive the content that was selected based on the first device ID. The browser's rendering of the selected content can be the step of rendering the content that the browser began to render at step 104. The browser can also store or cache the second device ID, which may contain the first device ID, at the client device. The browser can cache the second device ID as data in a data file associated with the content publishing device. For example, the cached copy of the second device ID can be stored in an HTTP cookie that is associated with the content publishing device. In some implementations, the cached copy of the second device ID can be stored into a partitioned portion of a cookie that is associated with the content publishing device. The partitioned portion of the cookie can be associated with the data processing system and the data processing system can access the partitioned portion of the cookie in third-party contexts. The browser can set a max-age or expiration time of the second device ID. The max-

age or expiration time can be a flag or data value that indicates to the browser when the second device ID expires or should be considered stale.

If at step 105 the browser determines the first device ID is not available, at step 111 the browser can determine if the second device ID is cached at the browser. For example, after a predetermined time storing the first device ID at step 102, the browser can partition or delete the data file (e.g., HTTP cookie) that contains the first device ID. The browser can determine if the second device ID associated with the first device ID is stored in a data file that is associated with the content publishing device. The browser can cache a copy of the second device ID during an earlier interaction with the content publishing device. If, for example, a request for content is transmitted to the content publishing device within 24 hours of receiving the first device ID (e.g., before the data file containing the first device ID is partitioned), the steps 103–110 can be completed to store a cached copy of the second device ID at the client device. After 24 hours, the first device ID may no longer be available in a third-party context, but the cached copy of the second device ID may be available and the steps 112–115 can occur. In some implementations, if the max-age or expiration time associated with the second device ID has passed, the browser can discard the second device ID and progress to step 116. If a valid (e.g., non-expired) cached copy of the second device ID exists at the browser, the browser can transmit the second device ID to the data processing system as part of a content request. The browser can append the second device ID (or an indication thereof) to a content request that is transmitted to the data processing system.

At step 112, the data processing system can receive the second device ID. At step 113, the data processing system can determine the first device ID based on at least the second device ID. In some implementations, the data processing system decrypts the second device ID to

determine the first device ID. In some implementations, the data processing system can use the second device ID (or decrypted data therefrom) as a key for a lookup table to retrieve the first device ID.

At step 114, the data processing system can select content based on at least the first device ID. As discussed above, the data processing system can use preferences or parameters associated with the first device ID to select the content. At step 115, the data processing system can transmit the selected content to the client device for rendering.

If at step 111 the browser determines that a copy of the second device ID is not available, the client device can transmit a content request to the data processing system. Since the first device ID and the second device ID are not available, the browser may not append a device ID to the content request to the data processing system.

At step 116, the data processing system can select content responsive to receiving the request. As described above, the content request in response to which the content is selected at step 116 may not include a device ID. The data processing system can transmit the content to the client device. At step 117 the browser can render the selected content.

ABSTRACT

The systems and methods described herein can enable the caching of data files associated with a first data processing system into the data files associated with a second data processing system. The systems and methods can enable the exchange of information between third-party entities that would be blocked by typical web browsers and other computing devices that prevent cross-domain data exchange. This allows for rich, multi-source content with embedded third-party content, mixed media, or other such content without sacrificing security of the client device and browser environment.